

LAB RECORD

23CSE111- Object Oriented Programming

***Submitted by***

CH.SC.U4CSE24130 - N.SARVAN KUMAR

# BACHELOR OF TECHNOLOGY

## IN

COMPUTER SCIENCE AND ENGINEERING

AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF COMPUTING

CHENNAI

March - 2025



**AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF COMPUTING, CHENNAI**

**BONAFIDE CERTIFICATE**

This is to certify that the Lab Record work for 23CSE111- Object Oriented Programming Subject submitted by ***CH.SC.U4CSE24130 - N.SARVAN KUMAR*** in **“Computer Science and Engineering”** is a Bonafide record of the work carried out under my guidance and supervision at Amrita School of Computing, Chennai.

This Lab examination held on / /2025

Internal Examiner 1 Internal Examiner 2

## INDEX

|  |  |  |
| --- | --- | --- |
| **S.NO** | **TITLE** | **PAGE.NO** |
| **UML DIAGRAM** | |  |
| 1. | **CAB BOOKING SYSTEM** |  |
|  | 1.a) Use Case Diagram |  |
|  | 1.b) Class Diagram |  |
|  | 1.c) Sequence Diagram |  |
|  | 1.d) State Diagram |  |
|  | 1.e) Object Diagram |  |
| 2. | **Student Attendance Management System** |  |
|  | 2.a) Use Case Diagram |  |
|  | 2.b) Class Diagram |  |
|  | 2.c) Sequence Diagram |  |
|  | 2.d) State Diagram |  |
|  | 2.e) Object Diagram |  |
| 3. | **BASIC JAVA PROGRAMS** |  |
|  | 3.a) Even Odd |  |
|  | 3.b) Factorial |  |
|  | 3.c) Greeting |  |
|  | 3.d) Largest Number |  |
|  | 3.e) Number Guessing Game |  |
|  | 3.f) Print Numbers using Loop |  |
|  | 3.g) Sum of Natural Numbers |  |
|  | 3.h) Circle Area |  |
|  | 3.i) Cube Volume |  |
|  | 3.j) Voting System |  |
|  | **INHERITANCE** |  |

|  |  |  |
| --- | --- | --- |
| 4. | **SINGLE INHERITANCE PROGRAMS** |  |
|  | 4.a) Book – Novel (Scanner) |  |
|  | 4.b) Animal – Dog (Scanner) |  |
| 5. | **MULTILEVEL INHERITANCE PROGRAMS** |  |
|  | 5.a) Appliance – Washing Machine (Scanner) |  |
|  | 5.b) Vehicle – Car (Scanner) |  |
| 6. | **HIERARCHICAL INHERITANCE PROGRAMS** |  |
|  | 6.a) Vehicle – Car (Scanner) |  |
|  | 6.b) Employee – Manager (Scanner) |  |
| 7. | **HYBRID INHERITANCE PROGRAMS** |  |
|  | 7.a) Animal – Bird (Scanner) |  |
|  | 7.b) Person – Employee (Scanner) |  |
|  | **POLYMORPHISM** |  |
| 8. | **CONSTRUCTOR PROGRAMS** |  |
|  | 8.a) Rectangle |  |
|  | 8.b) Student |  |
| 9. | **CONSTRUCTOR OVERLOADING PROGRAMS** |  |
|  | 9.a) Bank Account |  |
|  | 9.b) Employee |  |
| 10. | **METHOD OVERLOADING PROGRAMS** |  |
|  | 10.a) Adding Numbers |  |
|  | 10.b) Multiplying Numbers |  |
| 11. | **METHOD OVERRIDING PROGRAMS** |  |
|  | 11.a) Banking System |  |
|  | 11.b) Animal |  |
|  | **ABSTRACTION** |  |
| 12. | **INTERFACE PROGRAMS** |  |
|  | 12.a) Bank Account |  |
|  | 12.b) Employee Management |  |
|  | 12.c) Restauarant Order |  |
|  | 12.d) Shopping Cart |  |
| 13. | **ABSTRACT CLASS PROGRAMS** |  |
|  | 13.a) Animal |  |
|  | 13.b) Shape - Rectangle |  |
|  | 13.c) Vehicle - Car |  |
|  | 13.d) Calculator |  |
|  | **ENCAPSULATION** |  |
| 14. | **ENCAPSULATION PROGRAMS** |  |
|  | 14.a) Person – Getter & Setter |  |
|  | 14.b) Bank Account |  |
|  | 14.c) Employee – Getter & Setter |  |
|  | 14.d) Product – Getter & Setter |  |
| 15. | **PACKAGES PROGRAMS** |  |
|  | 15.a)User Defined Package - Banking |  |
|  | 15.b)User Defined Package - Geometry |  |
|  | 15.c)User Defined Package - Mathematics |  |
|  | 15.d)User Defined Package - Utilities |  |
|  | 15.e) Built – in Package – Array List |  |
|  | 15.f) Built – in Package – File, FileWriter etc., |  |
|  | 15.g) Built – in Package – LocalDate, LocalTime etc., |  |
|  | 15.h) Built – in Package – Executors etc., |  |
| 16. | **EXCEPTION HANDLING PROGRAMS** |  |
|  | 16.a) Array |  |
|  | 16.b) Custom Exception |  |
|  | 16.c) Divison |  |
|  | 16.d) Input |  |
| 17. | **FILE HANDLING PROGRAMS** |  |
|  | 17.a) Buffer Write |  |
|  | 17.b) File Copy |  |
|  | 17.c) File Read |  |
|  | 17.d) File Write |  |
| 18. | **RAPTOR FLOW CHARTS** |  |
|  | 18.a) Electricity Bill Generator |  |
|  | 18.b) Currency Converter |  |
|  | 18.c) Volume and Surface Area of Cube |  |
| 19. | **PROJECT – TASK MANAGER (OWN INTEREST)** |  |

#### UML DIAGRAM – 1 – CAB BOOKING SYSTEM

#### 1.a) CAB BOOKING SYSTEM Use Case Diagram

#### 

#### 1.b) CAB BOOKING SYSTEM Class Diagram

#### 

#### 1.c) CAB BOOKING SYSTEM Sequence Diagram

#### 

#### 1.d) CAB BOOKING SYSTEM State Diagram

#### 

#### 1.e) CAB BOOKING SYSTEM Object Diagram

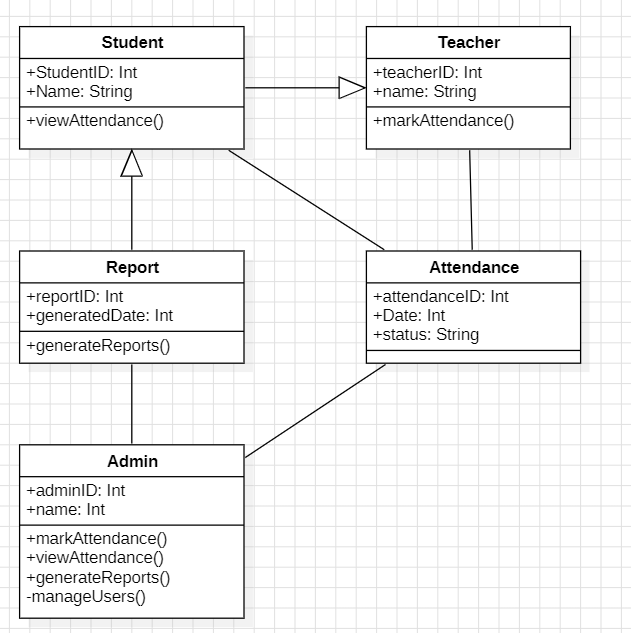
#### 

#### UML DIAGRAM – 2 – STUDENT ATTENDANCE MANAGEMENT SYSTEM

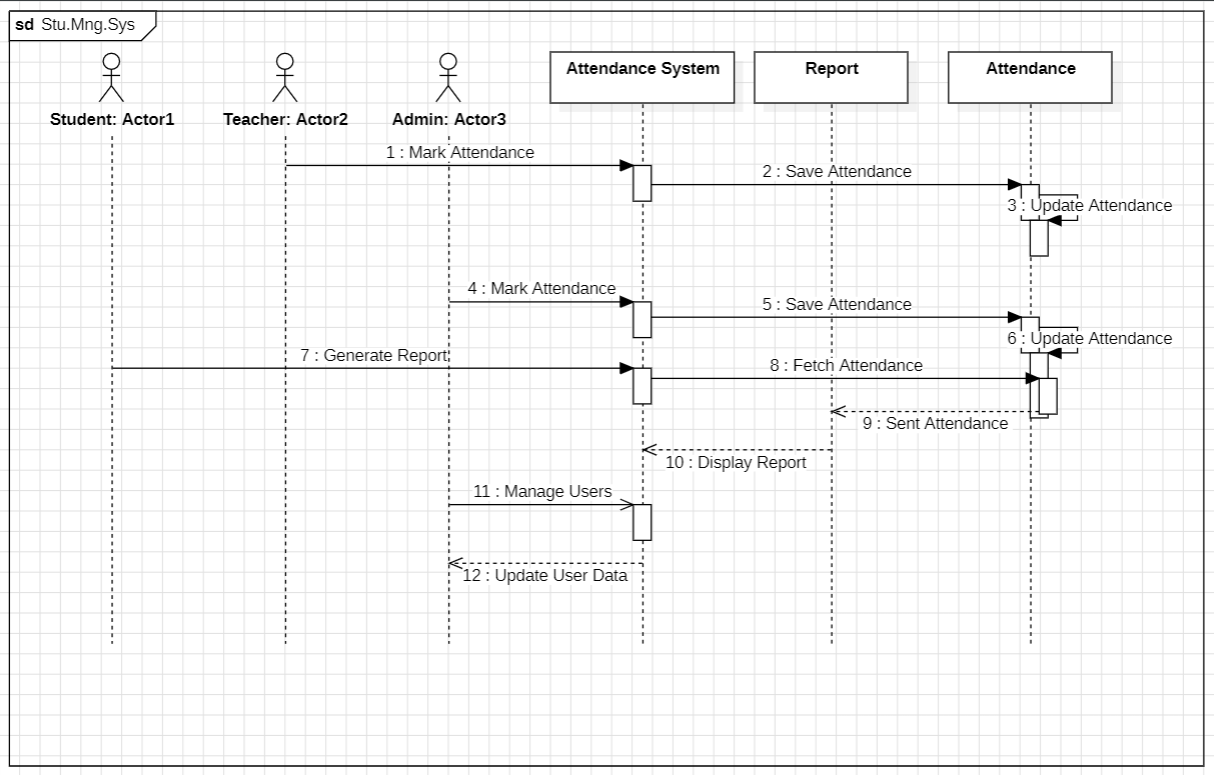
#### 2.a) STUDENT ATTENDANCE MANAGEMENT SYSTEM Use Case Diagram

#### 

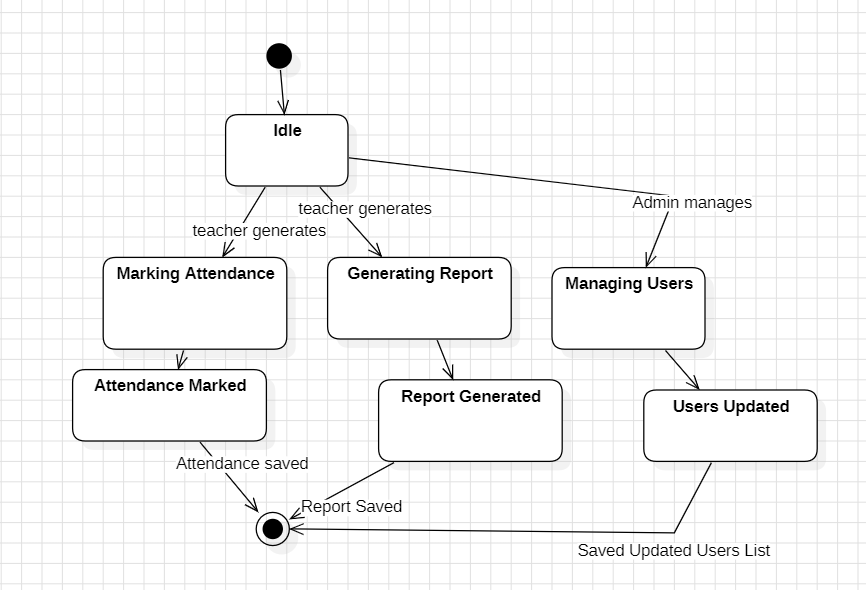
#### 2.b) STUDENT ATTENDANCE MANAGEMENT SYSTEM Class Diagram

****

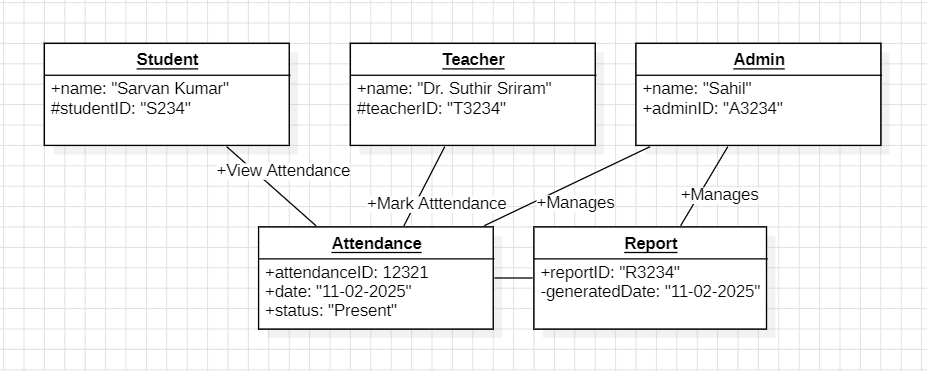
#### 2.c) STUDENT ATTENDANCE MANAGEMENT SYSTEM Sequence Diagram

****

#### 2.d) STUDENT ATTENDANCE MANAGEMENT SYSTEM State Diagram

****

#### 2.e) STUDENT ATTENDANCE MANAGEMENT SYSTEM Object Diagram

****

**JAVA BASIC PROGRAMS**

1. **Even Odd**

**Code:**

import java.util.Scanner;

public class EvenOdd {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter a number: ");

int num = sc.nextInt();

if (num % 2 == 0) {

System.out.println(num + " is even.");

} else {

System.out.println(num + " is odd.");

}

sc.close();

}

}

**Output:**

****

1. **Factorial**

**Code:**

import java.util.Scanner;

public class Factorial {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter a number: ");

int n = sc.nextInt();

long fact = 1;

for (int i = 1; i <= n; i++) {

fact \*= i;

}

System.out.println("Factorial of " + n + " is: " + fact);

sc.close();

}

}

**Output:**

****

1. **Greeting**

**Code:**

import java.util.Scanner;

public class Greeting {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter your name: ");

String name = scanner.nextLine();

System.out.println("Hello, " + name + "!");

scanner.close();

}

}

**Output:**

****

1. **Largest Number**

**Code:**

import java.util.Scanner;

public class LargestNumber {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter three numbers: ");

int a = sc.nextInt();

int b = sc.nextInt();

int c = sc.nextInt();

int max = (a > b) ? (a > c ? a : c) : (b > c ? b : c);

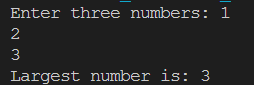
System.out.println("Largest number is: " + max);

sc.close();

}

}

**Output:**

****

1. **Number Guessing Game**

**Code:**

import java.util.Scanner;

import java.util.Random;

public class NumberGuessingGame {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Random rand = new Random();

int numberToGuess = rand.nextInt(100) + 1; // Random number between 1 and 100

int attempts = 0, guess;

System.out.println("Guess the number (between 1 and 100): ");

do {

System.out.print("Enter your guess: ");

guess = sc.nextInt();

attempts++;

if (guess < numberToGuess) {

System.out.println("Too low! Try again.");

} else if (guess > numberToGuess) {

System.out.println("Too high! Try again.");

} else {

System.out.println("Congratulations! You guessed it in " + attempts + " attempts.");

}

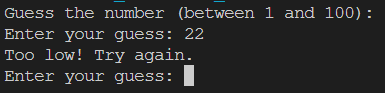
} while (guess != numberToGuess);

sc.close();

}

}

**Output:**

****

1. **Print Natural Numbers**

**Code:**

import java.util.Scanner;

public class PrintNumbers {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter a number (N): ");

int n = sc.nextInt();

for (int i = 1; i <= n; i++) {

System.out.print(i + " ");

}

sc.close();

}

}

**Output:**

****

1. **Sum of Natural Numbers**

**Code:**

import java.util.Scanner;

public class SumNaturalNumbers {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter a number (N): ");

int n = sc.nextInt();

int sum = 0, i = 1;

while (i <= n) {

sum += i;

i++;

}

System.out.println("Sum of first " + n + " natural numbers is: " + sum);

sc.close();

}

}

**Output:**

****

1. **Circle**

**Code:**

import java.util.Scanner;

public class circle {

public static void main(String[] args) {

Scanner tool = new Scanner(System.in);

System.out.println("Enter the Radius of Circle: ");

double radius = tool.nextInt();

double area = (3.14\*radius\*radius);

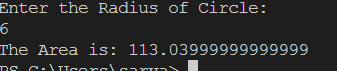
System.out.println("The Area is: "+ area);

tool.close();

}

}

**Output:**

****

1. **Cube**

**Code:**

import java.util.\*;

public class cubeVol{

public static void main(String[] args) {

Scanner tool = new Scanner(System.in);

System.out.println("Enter the length of cube: ");

double length = tool.nextDouble();

double volume = (length\*length\*length);

System.out.println("The volume of Cube is: " +volume);

tool.close();

}

}

**Output:**

****

1. **Voting**

**Code:**

import java.util.\*;

public class voting{

public static void main(String[] args){

Scanner tool = new Scanner(System.in);

System.out.println("Enter your age: ");

int age = tool.nextInt();

if (age>=18){

System.out.println("You Can Vote");

} else{

System.out.println("You Cannot Vote");

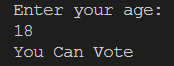
}

tool.close();

}

}

**Output:**

****

**JAVA SINGLE INHERITENCE PROGRAMS**

1. **Book – Novel**

**Code:**

import java.util.Scanner;

class Book {

String title;

void getTitle(Scanner sc) {

System.out.print("Enter book title: ");

title = sc.nextLine();

}

void displayTitle() {

System.out.println("Book Title: " + title);

}

}

class Novel extends Book {

String author;

void getAuthor(Scanner sc) {

System.out.print("Enter author name: ");

author = sc.nextLine();

}

void displayAuthor() {

System.out.println("Author: " + author);

}

}

public class SingleInheritanceExample1 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Novel novel = new Novel();

novel.getTitle(sc);

novel.getAuthor(sc);

novel.displayTitle();

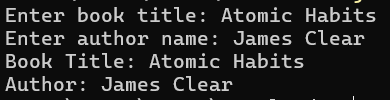
novel.displayAuthor();

sc.close();

}

}

**Output:**

****

1. **Animal - Dog**

**Code:**

import java.util.Scanner;

class Animal {

String name;

Animal(String name) {

this.name = name;

}

void showAnimalDetails() {

System.out.println("Animal Name: " + name);

}

}

class Dog extends Animal {

String breed;

Dog(String name, String breed) {

super(name);

this.breed = breed;

}

void showDogDetails() {

showAnimalDetails();

System.out.println("Breed: " + breed);

}

}

public class SingleInheritanceExample2 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter Dog Name: ");

String name = sc.nextLine();

System.out.print("Enter Breed: ");

String breed = sc.nextLine();

Dog dog = new Dog(name, breed);

System.out.println("\nDog Details:");

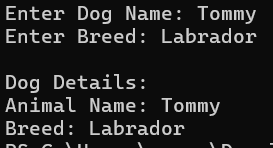
dog.showDogDetails();

sc.close();

}

}

**Output:**

****

**JAVA MULTI LEVEL INHERITENCE PROGRAMS**

1. **Appliances**

**Code:**

import java.util.Scanner;

class Appliance {

String brand;

void getBrand(Scanner sc) {

System.out.print("Enter appliance brand: ");

brand = sc.nextLine();

}

void displayBrand() {

System.out.println("Brand: " + brand);

}

}

class WashingMachine extends Appliance {

void washClothes() {

System.out.println(brand + " washing machine is washing clothes.");

}

}

class SmartWashingMachine extends WashingMachine {

void smartWash() {

System.out.println(brand + " smart washing machine optimizes water usage.");

}

}

public class MultilevelInheritanceExample1 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

SmartWashingMachine machine = new SmartWashingMachine();

machine.getBrand(sc);

machine.displayBrand();

machine.washClothes();

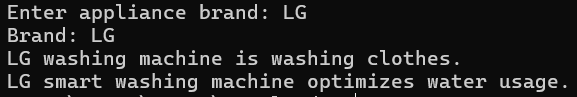
machine.smartWash();

sc.close();

}

}

**Output:**

****

1. **Vehcle**

**Code:**

import java.util.Scanner;

class Vehicle {

String brand;

int wheels;

Vehicle(String brand, int wheels) {

this.brand = brand;

this.wheels = wheels;

}

void showVehicleDetails() {

System.out.println("Brand: " + brand);

System.out.println("Number of Wheels: " + wheels);

}

}

class Car extends Vehicle {

String fuelType;

Car(String brand, int wheels, String fuelType) {

super(brand, wheels);

this.fuelType = fuelType;

}

void showCarDetails() {

showVehicleDetails();

System.out.println("Fuel Type: " + fuelType);

}

}

class SportsCar extends Car {

int maxSpeed;

SportsCar(String brand, int wheels, String fuelType, int maxSpeed) {

super(brand, wheels, fuelType);

this.maxSpeed = maxSpeed;

}

void showSportsCarDetails() {

showCarDetails();

System.out.println("Max Speed: " + maxSpeed + " km/h");

}

}

public class MultilevelInheritanceExample2 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter Car Brand: ");

String brand = sc.nextLine();

System.out.print("Enter Number of Wheels: ");

int wheels = sc.nextInt();

sc.nextLine();

System.out.print("Enter Fuel Type: ");

String fuelType = sc.nextLine();

System.out.print("Enter Max Speed: ");

int maxSpeed = sc.nextInt();

SportsCar sportsCar = new SportsCar(brand, wheels, fuelType, maxSpeed);

System.out.println("\nSports Car Details:");

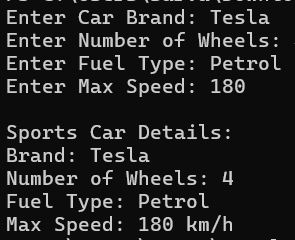
sportsCar.showSportsCarDetails();

sc.close();

}

}

**Output:**

****

**JAVA HIERARCHIAL INHERITENCE PROGRAMS**

1. **Vehicle**

**Code:**

import java.util.Scanner;

class Vehicle {

String name;

void getName(Scanner sc) {

System.out.print("Enter vehicle name: ");

name = sc.nextLine();

}

void displayName() {

System.out.println("Vehicle Name: " + name);

}

}

class Car extends Vehicle {

void drive() {

System.out.println(name + " is driving.");

}

}

class Bike extends Vehicle {

void ride() {

System.out.println(name + " is riding.");

}

}

public class HierarchialInheritanceExample1 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Car car = new Car();

car.getName(sc);

car.displayName();

car.drive();

Bike bike = new Bike();

bike.getName(sc);

bike.displayName();

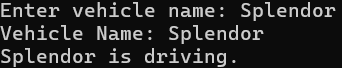
bike.ride();

sc.close();

}

}

**Output:**

****

1. **Employee**

**Code:**

import java.util.Scanner;

class Employee {

String name;

double salary;

Employee(String name, double salary) {

this.name = name;

this.salary = salary;

}

void display() {

System.out.println("Employee Name: " + name);

System.out.println("Salary: $" + salary);

}

}

class Manager extends Employee {

int teamSize;

Manager(String name, double salary, int teamSize) {

super(name, salary);

this.teamSize = teamSize;

}

void showManagerDetails() {

display();

System.out.println("Team Size: " + teamSize);

}

}

class Developer extends Employee {

String programmingLanguage;

Developer(String name, double salary, String programmingLanguage) {

super(name, salary);

this.programmingLanguage = programmingLanguage;

}

void showDeveloperDetails() {

display();

System.out.println("Programming Language: " + programmingLanguage);

}

}

public class HierarchialInheritenceExample2 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter Manager Name: ");

String managerName = sc.nextLine();

System.out.print("Enter Manager Salary: ");

double managerSalary = sc.nextDouble();

System.out.print("Enter Team Size: ");

int teamSize = sc.nextInt();

sc.nextLine();

Manager manager = new Manager(managerName, managerSalary, teamSize);

System.out.print("Enter Developer Name: ");

String developerName = sc.nextLine();

System.out.print("Enter Developer Salary: ");

double developerSalary = sc.nextDouble();

sc.nextLine();

System.out.print("Enter Programming Language: ");

String programmingLanguage = sc.nextLine();

Developer developer = new Developer(developerName, developerSalary, programmingLanguage);

System.out.println("\nManager Details:");

manager.showManagerDetails();

System.out.println("\nDeveloper Details:");

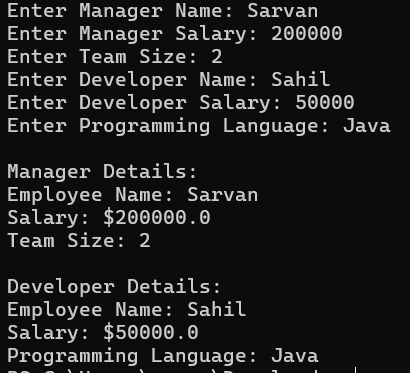
developer.showDeveloperDetails();

sc.close();

}

}

**Output:**

****

**JAVA HYBRID INHERITENCE PROGRAMS**

1. **Vehicle**

**Code:**

import java.util.Scanner;

class Animal {

String species;

void getSpecies(Scanner sc) {

System.out.print("Enter species: ");

species = sc.nextLine();

}

void displaySpecies() {

System.out.println("Species: " + species);

}

}

class Bird extends Animal {

void fly() {

System.out.println(species + " can fly.");

}

}

class Fish extends Animal {

void swim() {

System.out.println(species + " can swim.");

}

}

class Penguin extends Bird {

void cannotFly() {

System.out.println(species + " is a bird but cannot fly.");

}

}

public class HybridInheritanceExample1 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Penguin penguin = new Penguin();

penguin.getSpecies(sc);

penguin.displaySpecies();

penguin.fly();

penguin.cannotFly();

Fish fish = new Fish();

fish.getSpecies(sc);

fish.displaySpecies();

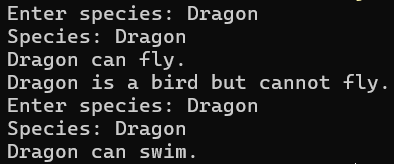
fish.swim();

sc.close();

}

}

**Output:**

****

1. **Person**

**Code:**

import java.util.Scanner;

class Person {

String name;

int age;

Person(String name, int age) {

this.name = name;

this.age = age;

}

void showDetails() {

System.out.println("Name: " + name);

System.out.println("Age: " + age);

}

}

class Employee extends Person {

int empID;

Employee(String name, int age, int empID) {

super(name, age);

this.empID = empID;

}

void showEmployeeDetails() {

showDetails();

System.out.println("Employee ID: " + empID);

}

}

class Professor extends Employee {

String subject;

Professor(String name, int age, int empID, String subject) {

super(name, age, empID);

this.subject = subject;

}

void showProfessorDetails() {

showEmployeeDetails();

System.out.println("Subject: " + subject);

}

}

class Administrator extends Employee {

String department;

Administrator(String name, int age, int empID, String department) {

super(name, age, empID);

this.department = department;

}

void showAdministratorDetails() {

showEmployeeDetails();

System.out.println("Department: " + department);

}

}

class HOD extends Professor {

String researchArea;

HOD(String name, int age, int empID, String subject, String researchArea) {

super(name, age, empID, subject);

this.researchArea = researchArea;

}

void showHODDetails() {

showProfessorDetails();

System.out.println("Research Area: " + researchArea);

}

}

public class HybridInheritenceExample2 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter Professor Name: ");

String profName = sc.nextLine();

System.out.print("Enter Age: ");

int profAge = sc.nextInt();

System.out.print("Enter Employee ID: ");

int profID = sc.nextInt();

sc.nextLine();

System.out.print("Enter Subject: ");

String subject = sc.nextLine();

Professor professor = new Professor(profName, profAge, profID, subject);

System.out.print("\nEnter Administrator Name: ");

String adminName = sc.nextLine();

System.out.print("Enter Age: ");

int adminAge = sc.nextInt();

System.out.print("Enter Employee ID: ");

int adminID = sc.nextInt();

sc.nextLine();

System.out.print("Enter Department: ");

String department = sc.nextLine();

Administrator administrator = new Administrator(adminName, adminAge, adminID, department);

System.out.print("\nEnter HOD Name: ");

String hodName = sc.nextLine();

System.out.print("Enter Age: ");

int hodAge = sc.nextInt();

System.out.print("Enter Employee ID: ");

int hodID = sc.nextInt();

sc.nextLine();

System.out.print("Enter Subject: ");

String hodSubject = sc.nextLine();

System.out.print("Enter Research Area: ");

String researchArea = sc.nextLine();

HOD hod = new HOD(hodName, hodAge, hodID, hodSubject, researchArea);

System.out.println("\nProfessor Details:");

professor.showProfessorDetails();

System.out.println("\nAdministrator Details:");

administrator.showAdministratorDetails();

System.out.println("\nHOD Details:");

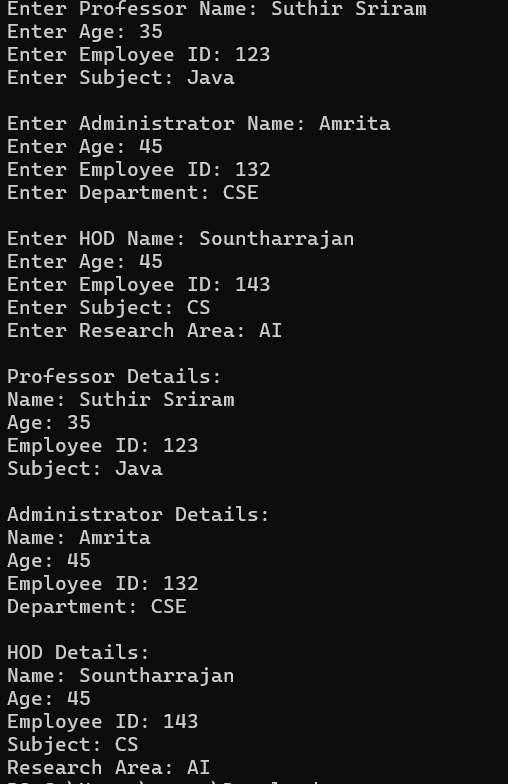
hod.showHODDetails();

sc.close();

}

}

**Output:**

****

**JAVA POLYMORPHISM PROGRAMS**

**CONSTRUCTORS**

1. **Rectangle**

**Code:**

import java.util.Scanner;

class Rectangle {

double length, width;

Rectangle(double length, double width) {

this.length = length;

this.width = width;

}

double calculateArea() {

return length \* width;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter Length of the Rectangle: ");

double length = sc.nextDouble();

System.out.print("Enter Width of the Rectangle: ");

double width = sc.nextDouble();

Rectangle rect = new Rectangle(length, width);

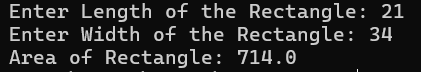
System.out.println("Area of Rectangle: " + rect.calculateArea());

sc.close();

}

}

**Output:**

****

1. **Student**

**Code:**

import java.util.Scanner;

class Student {

String name;

int age;

double marks;

Student(String name, int age, double marks) {

this.name = name;

this.age = age;

this.marks = marks;

}

void display() {

System.out.println("Student Name: " + name);

System.out.println("Age: " + age);

System.out.println("Marks: " + marks);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter Student Name: ");

String name = sc.nextLine();

System.out.print("Enter Age: ");

int age = sc.nextInt();

System.out.print("Enter Marks: ");

double marks = sc.nextDouble();

Student s = new Student(name, age, marks);

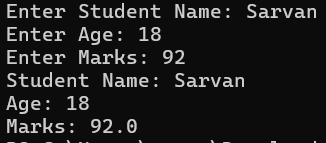
s.display();

sc.close();

}

}

**Output:**

****

**CONSTRUCTOR OVERLOADING**

1. **Bank Account**

**Code:**

import java.util.Scanner;

class BankAccount {

String accountHolder;

double balance;

BankAccount() {

this.accountHolder = "Unknown";

this.balance = 0.0;

}

BankAccount(String accountHolder) {

this.accountHolder = accountHolder;

this.balance = 5000;

}

BankAccount(String accountHolder, double balance) {

this.accountHolder = accountHolder;

this.balance = balance;

}

void display() {

System.out.println("Account Holder: " + accountHolder);

System.out.println("Balance: $" + balance);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

BankAccount acc1 = new BankAccount();

System.out.print("Enter Account Holder Name: ");

String name = sc.nextLine();

BankAccount acc2 = new BankAccount(name);

System.out.print("Enter Initial Deposit: ");

double deposit = sc.nextDouble();

BankAccount acc3 = new BankAccount(name, deposit);

System.out.println("\nDefault Account:");

acc1.display();

System.out.println("\nAccount with Name:");

acc2.display();

System.out.println("\nAccount with Initial Deposit:");

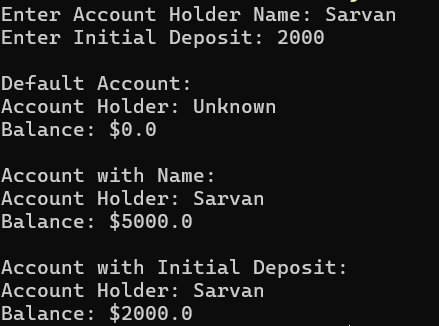
acc3.display();

sc.close();

}

}

**Output:**

****

1. **Employee**

**Code:**

import java.util.Scanner;

class Employee {

String name;

int age;

double salary;

Employee() {

this.name = "Unknown";

this.age = 0;

this.salary = 0.0;

}

Employee(String name, int age) {

this.name = name;

this.age = age;

this.salary = 30000;

}

Employee(String name, int age, double salary) {

this.name = name;

this.age = age;

this.salary = salary;

}

void display() {

System.out.println("Employee Name: " + name);

System.out.println("Age: " + age);

System.out.println("Salary: $" + salary);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Employee e1 = new Employee();

System.out.print("Enter Employee Name: ");

String name = sc.nextLine();

System.out.print("Enter Age: ");

int age = sc.nextInt();

Employee e2 = new Employee(name, age);

System.out.print("Enter Salary: ");

double salary = sc.nextDouble();

Employee e3 = new Employee(name, age, salary);

System.out.println("\nDefault Employee:");

e1.display();

System.out.println("\nEmployee with Name & Age:");

e2.display();

System.out.println("\nEmployee with All Details:");

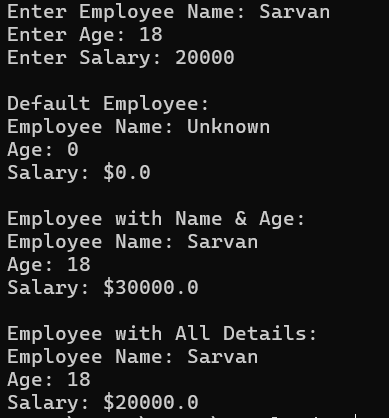
e3.display();

sc.close();

}

}

**Output:**

****

**METHOD OVERLOADING**

1. **Addition**

**Code:**

import java.util.Scanner;

public class MethodOverloadingExample1 {

static int add(int a, int b) {

return a + b;

}

static int add(int a, int b, int c) {

return a + b + c;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter two numbers: ");

int num1 = scanner.nextInt();

int num2 = scanner.nextInt();

System.out.print("Enter a third number (optional, enter 0 if not needed): ");

int num3 = scanner.nextInt();

if (num3 == 0) {

System.out.println("Sum of two numbers: " + add(num1, num2));

} else {

System.out.println("Sum of three numbers: " + add(num1, num2, num3));

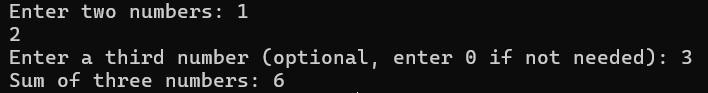
}

scanner.close();

}

}

**Output:**

****

1. **Multiplication**

**Code:**

import java.util.Scanner;

public class MethodOverloadingExample2 {

static int multiply(int a, int b) {

return a \* b;

}

static double multiply(double a, double b) {

return a \* b;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter first number (integer or decimal): ");

if (scanner.hasNextInt()) {

int num1 = scanner.nextInt();

System.out.print("Enter second number (integer): ");

int num2 = scanner.nextInt();

System.out.println("Product of two integers: " + multiply(num1, num2));

} else {

double num1 = scanner.nextDouble();

System.out.print("Enter second number (decimal): ");

double num2 = scanner.nextDouble();

System.out.println("Product of two doubles: " + multiply(num1, num2));

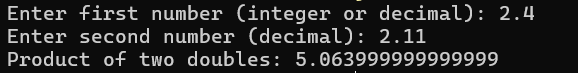
}

scanner.close();

}

}

**Output:**

****

**METHOD OVERRIDING**

1. **Banking**

**Code:**

import java.util.Scanner;

class Bank {

double getRateOfInterest() {

return 0;

}

}

class SBI extends Bank {

@Override

double getRateOfInterest() {

return 5.5;

}

}

class HDFC extends Bank {

@Override

double getRateOfInterest() {

return 6.0;

}

}

public class MethodOverridingExample1 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter bank name (SBI/HDFC): ");

String bankName = scanner.nextLine();

Bank bank;

if (bankName.equalsIgnoreCase("SBI")) {

bank = new SBI();

} else if (bankName.equalsIgnoreCase("HDFC")) {

bank = new HDFC();

} else {

System.out.println("Invalid bank name. Defaulting to Bank.");

bank = new Bank();

}

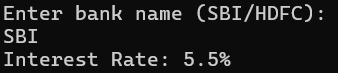
System.out.println("Interest Rate: " + bank.getRateOfInterest() + "%");

scanner.close();

}

}

**Output:**

****

1. **Animal**

**Code:**

import java.util.Scanner;

class Animal {

void makeSound() {

System.out.println("Some generic animal sound...");

}

}

class Dog extends Animal {

@Override

void makeSound() {

System.out.println("Dog barks: Bow Bow!");

}

}

class Cat extends Animal {

@Override

void makeSound() {

System.out.println("Cat meows: Meow Meow!");

}

}

public class MethodOverridingExample2 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter animal type (Dog/Cat): ");

String animalType = scanner.nextLine();

Animal animal;

if (animalType.equalsIgnoreCase("Dog")) {

animal = new Dog();

} else if (animalType.equalsIgnoreCase("Cat")) {

animal = new Cat();

} else {

System.out.println("Unknown animal. Defaulting to generic animal.");

animal = new Animal();

}

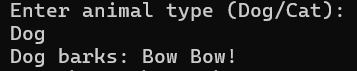
animal.makeSound();

scanner.close();

}

}

**Output:**

****

**USING INTERFACE**

1. **Banking**

**Code:**

import java.util.Scanner;

interface Bank {

void deposit(double amount);

void withdraw(double amount);

}

class SavingsAccount implements Bank {

double balance;

SavingsAccount(double balance) {

this.balance = balance;

}

public void deposit(double amount) {

balance += amount;

System.out.println("Deposited: " + amount);

}

public void withdraw(double amount) {

if (amount <= balance) {

balance -= amount;

System.out.println("Withdrawn: " + amount);

} else {

System.out.println("Insufficient balance!");

}

}

void showBalance() {

System.out.println("Current Balance: " + balance);

}

}

public class BankAccount {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter initial balance: ");

double balance = sc.nextDouble();

SavingsAccount acc = new SavingsAccount(balance);

System.out.print("Enter amount to deposit: ");

acc.deposit(sc.nextDouble());

System.out.print("Enter amount to withdraw: ");

acc.withdraw(sc.nextDouble());

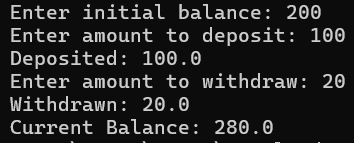
acc.showBalance();

sc.close();

}

}

**Output:**

****

1. **Employee Management**

**Code:**

import java.util.Scanner;

interface Employee {

void showEmployeeDetails();

}

class Manager implements Employee {

String name;

int salary;

Manager(String name, int salary) {

this.name = name;

this.salary = salary;

}

public void showEmployeeDetails() {

System.out.println("Manager Name: " + name);

System.out.println("Salary: " + salary);

}

}

public class EmployeeManagement {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter Manager Name: ");

String name = sc.nextLine();

System.out.print("Enter Salary: ");

int salary = sc.nextInt();

Employee emp = new Manager(name, salary);

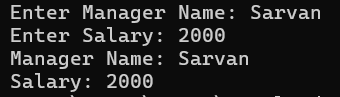
emp.showEmployeeDetails();

sc.close();

}

}

**Output:**

****

1. **Restaurant Order**

**Code:**

import java.util.Scanner;

interface Order {

void placeOrder();

}

class FoodOrder implements Order {

String item;

double price;

FoodOrder(String item, double price) {

this.item = item;

this.price = price;

}

public void placeOrder() {

System.out.println("Order placed for " + item + " - $" + price);

}

}

public class RestaurantOrder {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter Food Item: ");

String item = sc.nextLine();

System.out.print("Enter Price: ");

double price = sc.nextDouble();

Order order = new FoodOrder(item, price);

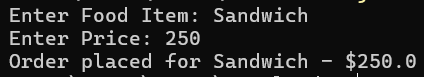
order.placeOrder();

sc.close();

}

}

**Output:**

****

1. **Shopping Cart**

**Code:**

import java.util.Scanner;

interface Product {

void display();

}

class Electronics implements Product {

String name;

double price;

Electronics(String name, double price) {

this.name = name;

this.price = price;

}

public void display() {

System.out.println("Electronics: " + name + " - $" + price);

}

}

public class ShoppingCart {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter product name: ");

String name = sc.nextLine();

System.out.print("Enter product price: ");

double price = sc.nextDouble();

Product prod = new Electronics(name, price);

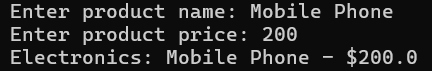
prod.display();

sc.close();

}

}

**Output:**

****

**JAVA ABSTRACTION PROGRAMS**

**USING ABSTRACT CLASS**

1. **Animal**

**Code:**

import java.util.Scanner;

abstract class Animal {

abstract void sound();

}

class Dog extends Animal {

void sound() {

System.out.println("Bark");

}

}

public class AbstractExample1 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Do you want to hear a dog's sound? (yes/no): ");

String input = scanner.nextLine();

if (input.equalsIgnoreCase("yes")) {

Animal myDog = new Dog();

myDog.sound();

} else {

System.out.println("No sound for now.");

}

scanner.close();

}

}

**Output:**

****

1. **Shape**

**Code:**

import java.util.Scanner;

abstract class Shape {

int x, y;

Shape(int x, int y) {

this.x = x;

this.y = y;

}

abstract void area();

}

class Rectangle extends Shape {

Rectangle(int x, int y) {

super(x, y);

}

void area() {

System.out.println("Area of rectangle: " + (x \* y));

}

}

public class AbstractExample2 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the length of the rectangle: ");

int length = scanner.nextInt();

System.out.print("Enter the width of the rectangle: ");

int width = scanner.nextInt();

Shape rectangle = new Rectangle(length, width);

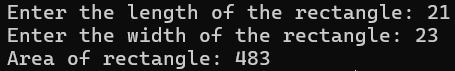
rectangle.area();

scanner.close();

}

}

**Output:**

****

1. **Car**

**Code:**

import java.util.Scanner;

abstract class Vehicle {

abstract void start();

abstract void stop();

void fuelType() {

System.out.println("Vehicle uses petrol.");

}

}

class Car extends Vehicle {

void start() {

System.out.println("Car is starting.");

}

void stop() {

System.out.println("Car is stopping.");

}

}

public class AbstractExample3 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Do you want to start the car? (yes/no): ");

String input = scanner.nextLine();

Vehicle myCar = new Car();

if (input.equalsIgnoreCase("yes")) {

myCar.start();

} else {

myCar.stop();

}

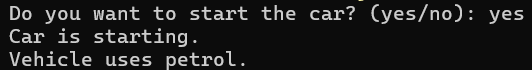
myCar.fuelType();

scanner.close();

}

}

**Output:**

****

1. **Calculator**

**Code:**

import java.util.Scanner;

abstract class Calculator {

abstract void add(int a, int b);

abstract void subtract(int a, int b);

}

class SimpleCalculator extends Calculator {

void add(int a, int b) {

System.out.println("Sum: " + (a + b));

}

void subtract(int a, int b) {

System.out.println("Difference: " + (a - b));

}

}

public class AbstractExample4 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the first number: ");

int num1 = scanner.nextInt();

System.out.print("Enter the second number: ");

int num2 = scanner.nextInt();

SimpleCalculator calculator = new SimpleCalculator();

calculator.add(num1, num2);

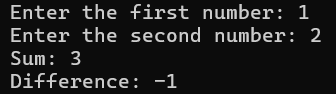
calculator.subtract(num1, num2);

scanner.close();

}

}

**Output:**

****

**JAVA ENCAPSULATION PROGRAMS**

1. **Person**

**Code:**

class Person {

private int age;

public int getAge() {

return age;

}

public void setAge(int age) {

if (age > 0) {

this.age = age;

} else {

System.out.println("Age must be positive!");

}

}

}

public class EncapsulationExample1 {

public static void main(String[] args) {

Person person = new Person();

person.setAge(25);

System.out.println("Person's age: " + person.getAge());

}

}

**Output:**

****

1. **Bank Account**

**Code:**

class BankAccount {

private double balance;

public double getBalance() {

return balance;

}

public void deposit(double amount) {

if (amount > 0) {

balance += amount;

System.out.println("Deposited: " + amount);

} else {

System.out.println("Deposit amount must be positive!");

}

}

public void withdraw(double amount) {

if (amount > 0 && amount <= balance) {

balance -= amount;

System.out.println("Withdrew: " + amount);

} else {

System.out.println("Invalid withdrawal amount!");

}

}

}

public class EncapsulationExample2 {

public static void main(String[] args) {

BankAccount account = new BankAccount();

account.deposit(500);

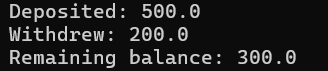
account.withdraw(200);

System.out.println("Remaining balance: " + account.getBalance());

}

}

**Output:**

****

1. **Employee**

**Code:**

class Employee {

private String name;

private double salary;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

if (salary >= 0) {

this.salary = salary;

} else {

System.out.println("Salary cannot be negative!");

}

}

}

public class EncapsulationExample3 {

public static void main(String[] args) {

Employee employee = new Employee();

employee.setName("John Doe");

employee.setSalary(45000);

System.out.println("Employee Name: " + employee.getName());

System.out.println("Employee Salary: " + employee.getSalary());

}

}

**Output:**

****

1. **Product**

**Code:**

class Product {

private String productCode;

private double price;

public String getProductCode() {

return productCode;

}

public void setProductCode(String productCode) {

this.productCode = productCode;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

if (price >= 0) {

this.price = price;

} else {

System.out.println("Price cannot be negative!");

}

}

}

public class EncapsulationExample4 {

public static void main(String[] args) {

Product product = new Product();

product.setProductCode("P12345");

product.setPrice(199.99);

System.out.println("Product Code: " + product.getProductCode());

System.out.println("Product Price: $" + product.getPrice());

}

}

**Output:**

****

**JAVA USER DEFINED PACKAGES**

1. **Banking**

**Account.java - Code:**

package banking;

public abstract class Account {

protected String accountNumber;

protected String accountHolder;

protected double balance;

public Account(String accountNumber, String accountHolder, double initialBalance) {

if (initialBalance < 0) {

throw new IllegalArgumentException("Initial balance cannot be negative");

}

this.accountNumber = accountNumber;

this.accountHolder = accountHolder;

this.balance = initialBalance;

}

public void deposit(double amount) {

if (amount <= 0) {

throw new IllegalArgumentException("Deposit amount must be positive");

}

balance += amount;

System.out.println("Deposited: $" + amount);

System.out.println("New Balance: $" + balance);

}

public abstract void withdraw(double amount);

public String getAccountNumber() {

return accountNumber;

}

public String getAccountHolder() {

return accountHolder;

}

public double getBalance() {

return balance;

}

public String getAccountDetails() {

return "Account Number: " + accountNumber +

"\nAccount Holder: " + accountHolder +

"\nCurrent Balance: $" + balance;

}

}

**CheckingAccount.java - Code:**

package banking;

public class CheckingAccount extends Account {

private double overdraftLimit;

public CheckingAccount(String accountNumber, String accountHolder,

double initialBalance, double overdraftLimit) {

super(accountNumber, accountHolder, initialBalance);

this.overdraftLimit = overdraftLimit;

}

@Override

public void withdraw(double amount) {

if (amount <= 0) {

throw new IllegalArgumentException("Withdrawal amount must be positive");

}

if (amount > (balance + overdraftLimit)) {

throw new IllegalStateException("Exceeds overdraft limit");

}

balance -= amount;

System.out.println("Withdrawn: $" + amount);

System.out.println("New Balance: $" + String.format("%.2f", balance));

if (balance < 0) {

System.out.println("Warning: Account is overdrawn!");

}

}

public double getOverdraftLimit() {

return overdraftLimit;

}

public void setOverdraftLimit(double overdraftLimit) {

this.overdraftLimit = overdraftLimit;

}

@Override

public String getAccountDetails() {

return super.getAccountDetails() +

"\nAccount Type: Checking" +

"\nOverdraft Limit: $" + overdraftLimit;

}

}

**SavingsAccount.java - Code:**

package banking;

public class SavingsAccount extends Account {

private double interestRate;

public SavingsAccount(String accountNumber, String accountHolder,

double initialBalance, double interestRate) {

super(accountNumber, accountHolder, initialBalance);

this.interestRate = interestRate;

}

@Override

public void withdraw(double amount) {

if (amount <= 0) {

throw new IllegalArgumentException("Withdrawal amount must be positive");

}

if (amount > balance) {

throw new IllegalStateException("Insufficient funds");

}

balance -= amount;

System.out.println("Withdrawn: $" + amount);

System.out.println("New Balance: $" + balance);

}

public void addInterest() {

double interest = balance \* interestRate / 100;

balance += interest;

System.out.println("Interest added: $" + String.format("%.2f", interest));

System.out.println("New Balance: $" + String.format("%.2f", balance));

}

public double getInterestRate() {

return interestRate;

}

public void setInterestRate(double interestRate) {

this.interestRate = interestRate;

}

@Override

public String getAccountDetails() {

return super.getAccountDetails() +

"\nAccount Type: Savings" +

"\nInterest Rate: " + interestRate + "%";

}

}

**Main Code:**

import banking.SavingsAccount;

import banking.CheckingAccount;

public class UserDefinedPackageExample4 {

public static void main(String[] args) {

System.out.println("Banking System Example:");

SavingsAccount savings = new SavingsAccount("SAV001", "John Doe", 1000.00, 2.5);

CheckingAccount checking = new CheckingAccount("CHK001", "Jane Smith", 500.00, 200.00);

System.out.println("\n--- Savings Account Details ---");

System.out.println(savings.getAccountDetails());

System.out.println("\n--- Checking Account Details ---");

System.out.println(checking.getAccountDetails());

System.out.println("\n--- Savings Account Transactions ---");

savings.deposit(500.00);

try {

savings.withdraw(200.00);

} catch (Exception e) {

System.out.println("Error: " + e.getMessage());

}

savings.addInterest();

System.out.println("\n--- Checking Account Transactions ---");

checking.deposit(100.00);

System.out.println("\nAttempt to withdraw within overdraft limit:");

try {

checking.withdraw(700.00);

} catch (Exception e) {

System.out.println("Error: " + e.getMessage());

}

System.out.println("\nAttempt to withdraw exceeding overdraft limit:");

try {

checking.withdraw(1000.00);

} catch (Exception e) {

System.out.println("Error: " + e.getMessage());

}

System.out.println("\nException Handling Examples:");

try {

savings.withdraw(-100.00);

} catch (IllegalArgumentException e) {

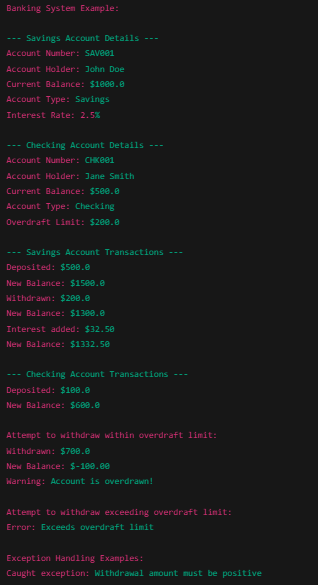
System.out.println("Caught exception: " + e.getMessage());

}

}

}

**Output:**

****

1. **Geometry**

**Circle.java - Code:**

package geometry;

public class Circle extends Shape {

private double radius;

public Circle(double radius) {

if (radius < 0) {

throw new IllegalArgumentException("Radius cannot be negative");

}

this.radius = radius;

}

@Override

public double area() {

return Math.PI \* radius \* radius;

}

@Override

public double perimeter() {

return 2 \* Math.PI \* radius;

}

public double getDiameter() {

return 2 \* radius;

}

public double getRadius() {

return radius;

}

public void setRadius(double radius) {

if (radius < 0) {

throw new IllegalArgumentException("Radius cannot be negative");

}

this.radius = radius;

}

}

**Rectangle.java - Code:**

package geometry;

public class Rectangle extends Shape {

private double length;

private double width;

public Rectangle(double length, double width) {

if (length < 0 || width < 0) {

throw new IllegalArgumentException("Dimensions cannot be negative");

}

this.length = length;

this.width = width;

}

@Override

public double area() {

return length \* width;

}

@Override

public double perimeter() {

return 2 \* (length + width);

}

public boolean isSquare() {

return length == width;

}

public double getDiagonal() {

return Math.sqrt(length \* length + width \* width);

}

// Getters and setters

public double getLength() {

return length;

}

public void setLength(double length) {

if (length < 0) {

throw new IllegalArgumentException("Length cannot be negative");

}

this.length = length;

}

public double getWidth() {

return width;

}

public void setWidth(double width) {

if (width < 0) {

throw new IllegalArgumentException("Width cannot be negative");

}

this.width = width;

}

}

**Shape.java - Code:**

package geometry;

public abstract class Shape {

public abstract double area();

public abstract double perimeter();

}

**Main Code:**

import geometry.Circle;

import geometry.Rectangle;

import geometry.Shape;

public class UserDefinedPackageExample3 {

public static void main(String[] args) {

System.out.println("Geometry Shapes Example:");

Circle circle = new Circle(5.0);

System.out.println("\nCircle with radius 5.0:");

System.out.println("Area: " + String.format("%.2f", circle.area()));

System.out.println("Perimeter: " + String.format("%.2f", circle.perimeter()));

System.out.println("Diameter: " + String.format("%.2f", circle.getDiameter()));

Rectangle rectangle = new Rectangle(4.0, 6.0);

System.out.println("\nRectangle with length 4.0 and width 6.0:");

System.out.println("Area: " + rectangle.area());

System.out.println("Perimeter: " + rectangle.perimeter());

System.out.println("Is Square? " + rectangle.isSquare());

System.out.println("Diagonal: " + String.format("%.2f", rectangle.getDiagonal()));

Rectangle square = new Rectangle(5.0, 5.0);

System.out.println("\nSquare with side 5.0:");

System.out.println("Area: " + square.area());

System.out.println("Perimeter: " + square.perimeter());

System.out.println("Is Square? " + square.isSquare());

System.out.println("\nDemonstrating polymorphism:");

Shape shape1 = new Circle(3.0);

Shape shape2 = new Rectangle(3.0, 4.0);

System.out.println("Shape1 (Circle) Area: " + String.format("%.2f", shape1.area()));

System.out.println("Shape2 (Rectangle) Area: " + shape2.area());

try {

Circle invalidCircle = new Circle(-1.0);

} catch (IllegalArgumentException e) {

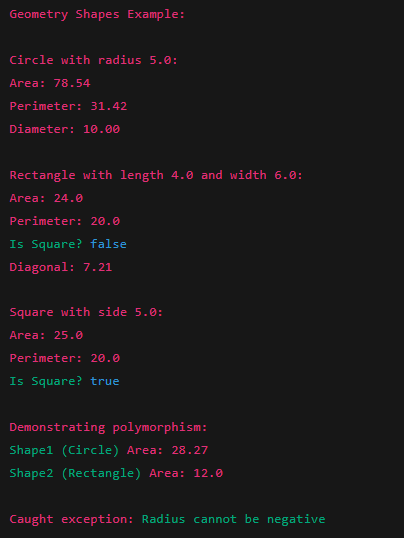
System.out.println("\nCaught exception: " + e.getMessage());

}

}

}

**Output:**

****

1. **Mathematics**

**MathOperations.java- Code:**

package mathematics;

public class MathOperations {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

public int multiply(int a, int b) {

return a \* b;

}

public double divide(int a, int b) {

if (b == 0) {

throw new ArithmeticException("Division by zero");

}

return (double) a / b;

}

public int power(int base, int exponent) {

int result = 1;

for (int i = 0; i < exponent; i++) {

result \*= base;

}

return result;

}

}

**Main Code:**

import mathematics.MathOperations;

public class UserDefinedPackageExample1 {

public static void main(String[] args) {

MathOperations math = new MathOperations();

int sum = math.add(10, 5);

int difference = math.subtract(20, 8);

int product = math.multiply(6, 7);

double quotient = math.divide(50, 4);

int powerResult = math.power(2, 8);

System.out.println("Mathematics Operations Example:");

System.out.println("Addition: 10 + 5 = " + sum);

System.out.println("Subtraction: 20 - 8 = " + difference);

System.out.println("Multiplication: 6 × 7 = " + product);

System.out.println("Division: 50 ÷ 4 = " + quotient);

System.out.println("Power: 2^8 = " + powerResult);

try {

double result = math.divide(10, 0);

} catch (ArithmeticException e) {

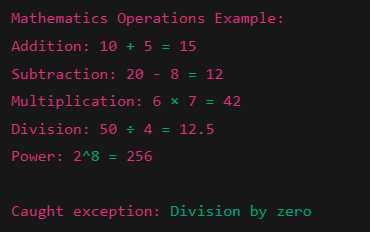
System.out.println("\nCaught exception: " + e.getMessage());

}

}

}

**Output:**



1. **Utilities**

**StringUtils.java- Code:**

package utilities;

public class StringUtils {

public String reverse(String input) {

StringBuilder sb = new StringBuilder(input);

return sb.reverse().toString();

}

public boolean isPalindrome(String input) {

String cleanInput = input.toLowerCase().replaceAll("[^a-z0-9]", "");

String reversed = reverse(cleanInput);

return cleanInput.equals(reversed);

}

public int countWords(String input) {

if (input == null || input.trim().isEmpty()) {

return 0;

}

return input.trim().split("\\s+").length;

}

public String capitalizeWords(String input) {

if (input == null || input.isEmpty()) {

return input;

}

StringBuilder result = new StringBuilder();

String[] words = input.split("\\s+");

for (String word : words) {

if (!word.isEmpty()) {

result.append(Character.toUpperCase(word.charAt(0)))

.append(word.substring(1).toLowerCase())

.append(" ");

}

}

return result.toString().trim();

}

}

**Main Code:**

import utilities.StringUtils;

public class UserDefinedPackageExample2 {

public static void main(String[] args) {

StringUtils utils = new StringUtils();

String text1 = "Hello, World!";

String text2 = "A man, a plan, a canal: Panama";

String text3 = "this is a test string for capitalization";

System.out.println("String Utilities Example:");

System.out.println("\nReverse:");

System.out.println("Original: " + text1);

System.out.println("Reversed: " + utils.reverse(text1));

System.out.println("\nPalindrome Check:");

System.out.println("\"" + text1 + "\" is a palindrome: " + utils.isPalindrome(text1));

System.out.println("\"" + text2 + "\" is a palindrome: " + utils.isPalindrome(text2));

System.out.println("\nWord Count:");

System.out.println("\"" + text1 + "\" has " + utils.countWords(text1) + " words");

System.out.println("\"" + text3 + "\" has " + utils.countWords(text3) + " words");

System.out.println("\nCapitalize Words:");

System.out.println("Original: " + text3);

System.out.println("Capitalized: " + utils.capitalizeWords(text3));

System.out.println("\nEmpty String Tests:");

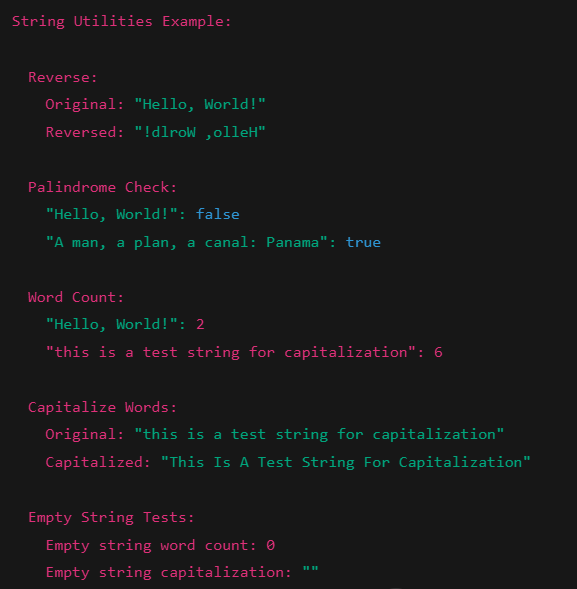
System.out.println("Empty string word count: " + utils.countWords(""));

System.out.println("Empty string capitalization: \"" + utils.capitalizeWords("") + "\"");

}

}

**Output:**

****

**JAVA BUILT-IN PACKAGES**

1. **Arraylist**

**Code:**

import java.util.ArrayList;

public class BuiltinPackageExample1 {

public static void main(String[] args) {

ArrayList<String> languages = new ArrayList<>();

languages.add("Java");

languages.add("Python");

languages.add("JavaScript");

languages.add("C++");

System.out.println("Programming Languages List:");

for (int i = 0; i < languages.size(); i++) {

System.out.println((i+1) + ". " + languages.get(i));

}

System.out.println("\nTotal languages: " + languages.size());

System.out.println("First language: " + languages.get(0));

System.out.println("Contains Python? " + languages.contains("Python"));

languages.remove("JavaScript");

System.out.println("\nAfter removing JavaScript:");

for (String lang : languages) {

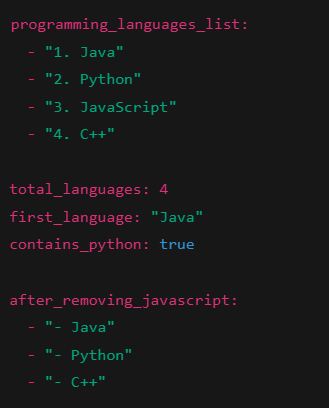
System.out.println("- " + lang);

}

}

}

**Output:**



1. **File**

**Code:**

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.io.BufferedReader;

import java.io.FileReader;

public class BuiltinPackageExample2 {

public static void main(String[] args) {

try {

File file = new File("sample.txt");

if (file.createNewFile()) {

System.out.println("File created: " + file.getName());

} else {

System.out.println("File already exists.");

}

FileWriter writer = new FileWriter(file);

writer.write("This is a demonstration of the java.io package.\n");

writer.write("File handling is an important part of Java programming.");

writer.close();

System.out.println("Successfully wrote to the file.");

System.out.println("\nContents of the file:");

BufferedReader reader = new BufferedReader(new FileReader(file));

String line;

while ((line = reader.readLine()) != null) {

System.out.println(line);

}

reader.close();

System.out.println("\nFile Information:");

System.out.println("Path: " + file.getAbsolutePath());

System.out.println("Size: " + file.length() + " bytes");

System.out.println("Can read: " + file.canRead());

System.out.println("Can write: " + file.canWrite());

} catch (IOException e) {

System.out.println("An error occurred.");

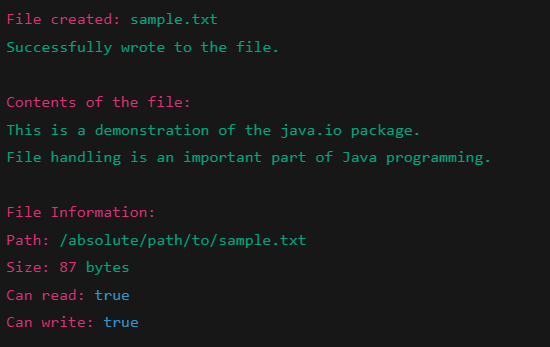
e.printStackTrace();

}

}

}

**Output:**



1. **LocalDate**

**Code:**

import java.time.LocalDate;

import java.time.LocalTime;

import java.time.LocalDateTime;

import java.time.ZonedDateTime;

import java.time.ZoneId;

import java.time.format.DateTimeFormatter;

import java.time.temporal.ChronoUnit;

public class BuiltinPackageExample3 {

public static void main(String[] args) {

LocalDate currentDate = LocalDate.now();

System.out.println("Current Date: " + currentDate);

LocalTime currentTime = LocalTime.now();

System.out.println("Current Time: " + currentTime);

LocalDateTime currentDateTime = LocalDateTime.now();

System.out.println("Current Date and Time: " + currentDateTime);

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("EEEE, MMMM dd, yyyy HH:mm:ss");

String formattedDateTime = currentDateTime.format(formatter);

System.out.println("\nFormatted Date and Time: " + formattedDateTime);

LocalDate futureDate = currentDate.plusDays(100);

LocalDate pastDate = currentDate.minusMonths(3);

System.out.println("\nDate after 100 days: " + futureDate);

System.out.println("Date 3 months ago: " + pastDate);

long daysBetween = ChronoUnit.DAYS.between(pastDate, futureDate);

System.out.println("Days between " + pastDate + " and " + futureDate + ": " + daysBetween);

ZonedDateTime nyTime = ZonedDateTime.now(ZoneId.of("America/New\_York"));

ZonedDateTime tokyoTime = ZonedDateTime.now(ZoneId.of("Asia/Tokyo"));

ZonedDateTime londonTime = ZonedDateTime.now(ZoneId.of("Europe/London"));

System.out.println("\nCurrent time in different zones:");

System.out.println("New York: " + nyTime.format(DateTimeFormatter.ofPattern("HH:mm:ss")));

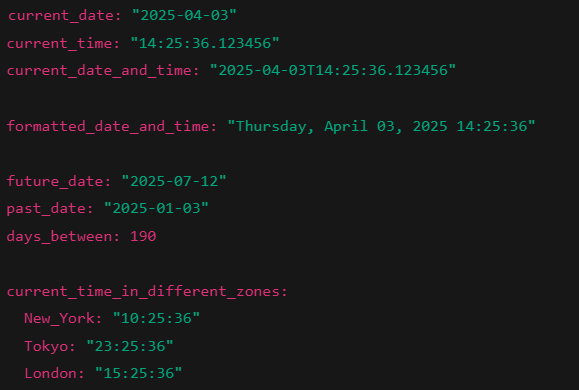
System.out.println("Tokyo: " + tokyoTime.format(DateTimeFormatter.ofPattern("HH:mm:ss")));

System.out.println("London: " + londonTime.format(DateTimeFormatter.ofPattern("HH:mm:ss")));

}

}

**Output:**



1. **Executors**

**Code:**

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.concurrent.TimeUnit;

import java.util.concurrent.atomic.AtomicInteger;

public class BuiltinPackageExample4 {

public static void main(String[] args) {

ExecutorService executor = Executors.newFixedThreadPool(3);

AtomicInteger counter = new AtomicInteger(0);

System.out.println("Starting concurrent tasks...");

for (int i = 1; i <= 5; i++) {

final int taskNum = i;

executor.submit(() -> {

try {

System.out.println("Task " + taskNum + " started by " +

Thread.currentThread().getName());

Thread.sleep((long)(Math.random() \* 1000));

int newCount = counter.incrementAndGet();

System.out.println("Task " + taskNum + " completed. Counter: " + newCount);

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

return null;

});

}

executor.shutdown();

try {

if (!executor.awaitTermination(5, TimeUnit.SECONDS)) {

System.out.println("Not all tasks completed in time.");

executor.shutdownNow();

} else {

System.out.println("\nAll tasks completed successfully.");

System.out.println("Final counter value: " + counter.get());

}

} catch (InterruptedException e) {

executor.shutdownNow();

Thread.currentThread().interrupt();

}

}

}

**Output:**



**JAVA EXCEPTION HANDLING PROGRAMS**

1. **Arrays**

**Code:**

import java.util.Scanner;

public class ArrayExample {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

String[] numbers = {"1", "2", "3"};

try {

System.out.print("Enter index: ");

int index = scanner.nextInt();

System.out.println(numbers[index]);

System.out.print("Enter a number: ");

int num = Integer.parseInt(scanner.next());

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Error: Array index out of bounds");

} catch (NumberFormatException e) {

System.out.println("Error: Invalid number format");

} finally {

System.out.println("Execution completed");

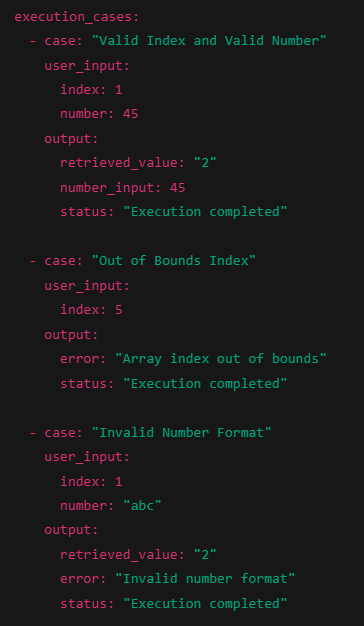
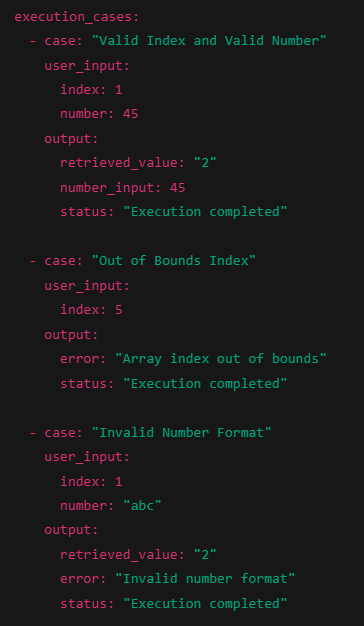
}

scanner.close();

}

}

**Possible Cases:**

1. **Custom Exception**

**Code:**

import java.util.Scanner;

class MyException extends Exception {

public MyException(String message) {

super(message);

}

}

public class CustomExceptionExample {

public static void checkAge(int age) throws MyException {

if (age < 18) {

throw new MyException("Age must be 18 or older");

}

System.out.println("Age is valid");

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Enter your age: ");

int age = scanner.nextInt();

checkAge(age);

} catch (MyException e) {

System.out.println("Caught custom exception: " + e.getMessage());

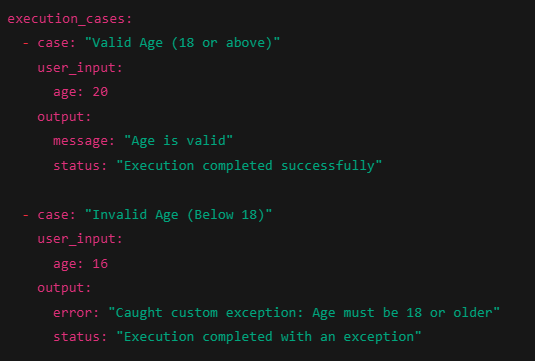
}

scanner.close();

}

}

**Possible Cases:**

****

1. **Divison**

**Code:**

import java.util.Scanner;

public class DivisionExample {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Enter first number: ");

int a = scanner.nextInt();

System.out.print("Enter second number: ");

int b = scanner.nextInt();

int result = a / b;

System.out.println("Result: " + result);

} catch (ArithmeticException e) {

System.out.println("Error: Division by zero is not allowed");

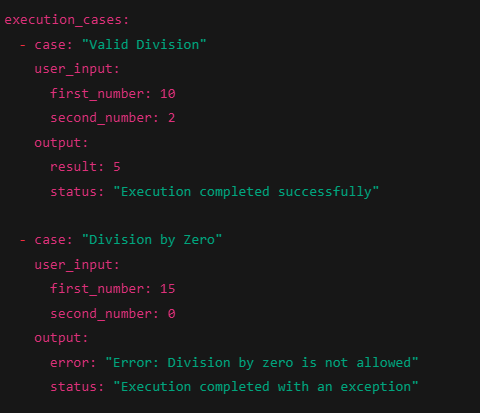
}

scanner.close();

}

}

**Possible Cases:**



1. **Input**

**Code:**

import java.util.Scanner;

public class InputExample {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Enter a number: ");

int num = scanner.nextInt();

System.out.println("You entered: " + num);

} catch (java.util.InputMismatchException e) {

System.out.println("Error: Please enter a valid integer");

} finally {

System.out.println("Program finished");

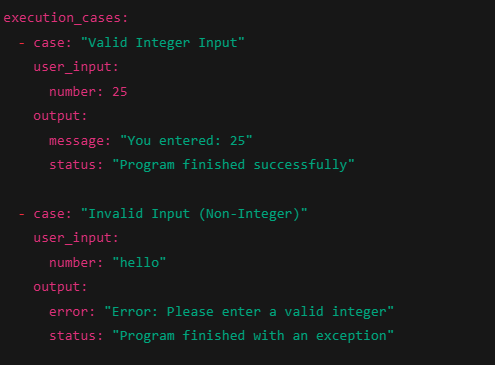
}

scanner.close();

}

}

**Possible Cases:**



**JAVA FILE HANDLING PROGRAMS**

1. **Buffer Write**

**Code:**

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.util.Scanner;

public class BufferedWriteExample {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try (BufferedWriter writer = new BufferedWriter(new FileWriter("buffered.txt"))) {

System.out.print("Enter number of lines: ");

int lines = scanner.nextInt();

scanner.nextLine();

for (int i = 0; i < lines; i++) {

System.out.print("Enter line " + (i+1) + ": ");

writer.write(scanner.nextLine());

writer.newLine();

}

System.out.println("File written successfully");

} catch (java.io.IOException e) {

System.out.println("Error writing to file: " + e.getMessage());

}

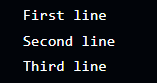
scanner.close();

}

}

**Output:**

**File Name: Buffered.txt**



1. **File Copy**

**Code:**

import java.io.FileReader;

import java.io.FileWriter;

import java.util.Scanner;

public class FileCopyExample {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Enter source file name: "); //source.txt

String source = scanner.nextLine();

System.out.print("Enter destination file name: "); //destination.txt

String destination = scanner.nextLine();

FileReader reader = new FileReader(source);

FileWriter writer = new FileWriter(destination);

int character;

while ((character = reader.read()) != -1) {

writer.write(character);

}

reader.close();

writer.close();

System.out.println("File copied successfully");

} catch (java.io.FileNotFoundException e) {

System.out.println("Error: File not found");

} catch (java.io.IOException e) {

System.out.println("Error during file operation: " + e.getMessage());

}

scanner.close();

}

}

**Output:**

**File Name: source.txt**

****

**File Name: destination.txt**

****

1. **File Read**

**Code:**

import java.io.FileReader;

import java.util.Scanner;

public class FileReadExample {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Enter file name to read: "); //read.txt

String fileName = scanner.nextLine();

FileReader reader = new FileReader(fileName);

int character;

while ((character = reader.read()) != -1) {

System.out.print((char)character);

}

reader.close();

} catch (java.io.FileNotFoundException e) {

System.out.println("Error: File not found");

} catch (java.io.IOException e) {

System.out.println("Error reading file: " + e.getMessage());

}

scanner.close();

}

}

**Output:**

**File Name: read.txt (Pre - Existing)**

****

1. **File Write**

**Code:**

import java.io.FileWriter;

import java.util.Scanner;

public class FileWriteExample {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

FileWriter writer = new FileWriter("output.txt");

System.out.print("Enter text to write: ");

String text = scanner.nextLine();

writer.write(text);

writer.close();

System.out.println("Successfully wrote to file");

} catch (java.io.IOException e) {

System.out.println("Error writing to file: " + e.getMessage());

}

scanner.close();

}

}

**Output:**

**File Name: output.txt**

****

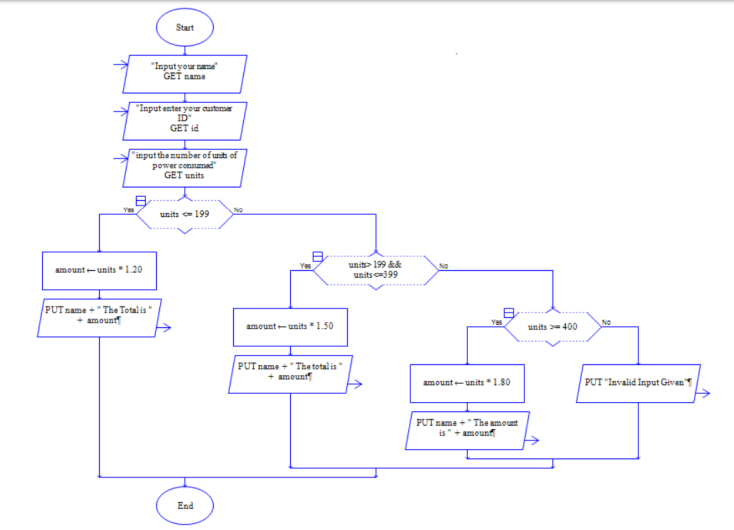
**RAPTOR FLOWCHARTS**

1. **Electricity Bill Generator**

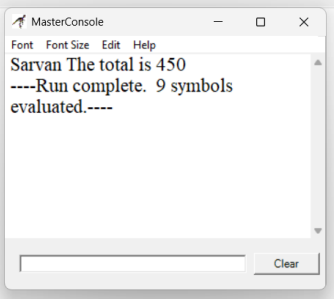
**For consumption up to 199 units, the charge per unit is Rs. 1.20.**

**For consumption between 200 and 399 units (inclusive), the charge per unit is Rs. 1.50.**

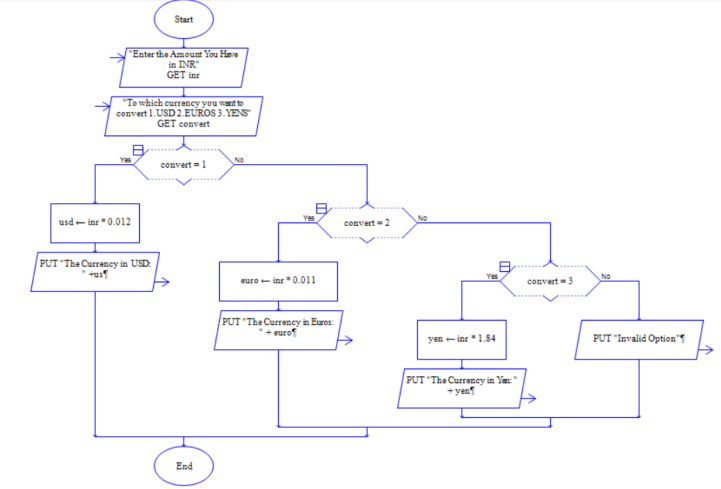
**For consumption above 400 units the charge per unit is Rs. 1.80.**

****

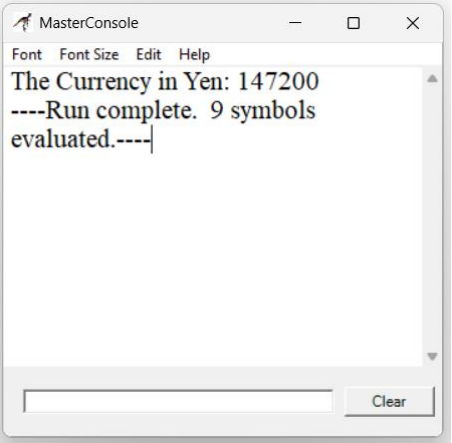
**Output:**

****

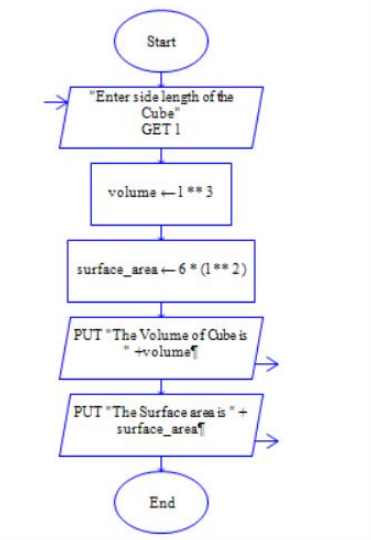
1. **Currency Converter to USD / EURO / YEN**

****

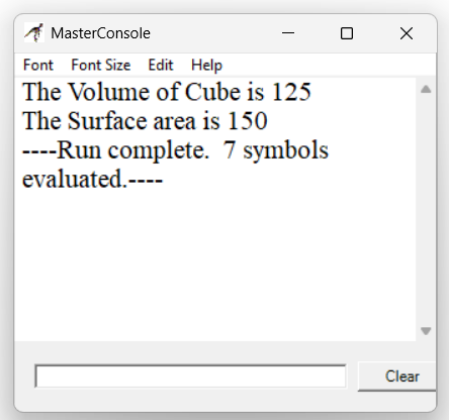
**Output:**

****

1. **Volume and surface area of Cube**

****

**Output:**

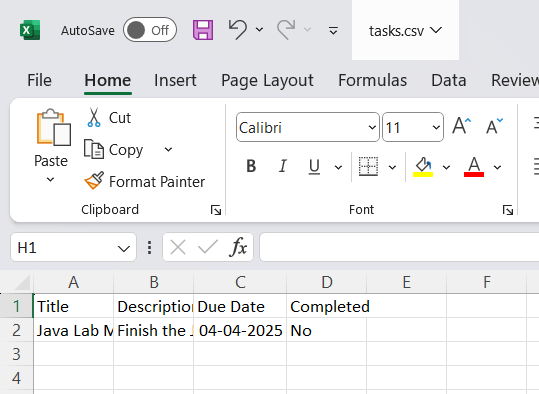
****

**OWN INTEREST PROJECT – TASK MANAGER**

**Description:**

The Task Manager is a simple console-based application built in Java for managing daily tasks efficiently. It provides users with an easy way to add, view, update, and delete tasks without requiring a database. Instead, all task data is stored in a file, ensuring persistence across multiple sessions. This lightweight application is ideal for individuals who need a quick and effective way to track their tasks without the complexity of database management.

**CSV/ Excel Output:**



**Code:**

import java.io.\*;

import java.util.\*;

class Task {

    String title;

    String description;

    String dueDate;

    boolean isCompleted;

    public Task(String title, String description, String dueDate, boolean isCompleted) {

        this.title = title;

        this.description = description;

        this.dueDate = dueDate;

        this.isCompleted = isCompleted;

    }

    @Override

    public String toString() {

        return (isCompleted ? "[✔] " : "[ ] ") + title + " (Due: " + dueDate + ") - " + description;

    }

}

public class TaskManager {

    private static final String FILE\_NAME = "tasks.txt";

    private static final List<Task> tasks = new ArrayList<>();

    public static void main(String[] args) {

        loadTasks();

        Scanner scanner = new Scanner(System.in);

        while (true) {

            System.out.println("\nTask Manager");

            System.out.println("1. Add Task");

            System.out.println("2. View Tasks");

            System.out.println("3. Mark Task as Completed");

            System.out.println("4. Delete Task");

            System.out.println("5. Save Tasks to CSV");

            System.out.println("6. Exit");

            System.out.print("Choose an option: ");

            int choice = scanner.nextInt();

            scanner.nextLine();

            switch (choice) {

                case 1 -> addTask(scanner);

                case 2 -> viewTasks();

                case 3 -> markTaskCompleted(scanner);

                case 4 -> deleteTask(scanner);

                case 5 -> saveTasksToCSV();

                case 6 -> {

                    saveTasks();

                    System.out.println("Exiting...");

                    return;

                }

                default -> System.out.println("Invalid option!");

            }

        }

    }

    private static void addTask(Scanner scanner) {

        System.out.print("Enter task title: ");

        String title = scanner.nextLine();

        System.out.print("Enter task description: ");

        String description = scanner.nextLine();

        System.out.print("Enter due date (YYYY-MM-DD): ");

        String dueDate = scanner.nextLine();

        tasks.add(new Task(title, description, dueDate, false));

        System.out.println("Task added successfully!");

    }

    private static void viewTasks() {

        if (tasks.isEmpty()) {

            System.out.println("No tasks available.");

            return;

        }

        for (int i = 0; i < tasks.size(); i++) {

            System.out.println((i + 1) + ". " + tasks.get(i));

        }

    }

    private static void markTaskCompleted(Scanner scanner) {

        viewTasks();

        if (tasks.isEmpty()) return;

        System.out.print("Enter task number to mark as completed: ");

        int index = scanner.nextInt() - 1;

        if (index >= 0 && index < tasks.size()) {

            tasks.get(index).isCompleted = true;

            System.out.println("Task marked as completed!");

        } else {

            System.out.println("Invalid task number.");

        }

    }

    private static void deleteTask(Scanner scanner) {

        viewTasks();

        if (tasks.isEmpty()) return;

        System.out.print("Enter task number to delete: ");

        int index = scanner.nextInt() - 1;

        if (index >= 0 && index < tasks.size()) {

            tasks.remove(index);

            System.out.println("Task deleted!");

        } else {

            System.out.println("Invalid task number.");

        }

    }

    private static void saveTasksToCSV() {

        try (FileWriter writer = new FileWriter("tasks.csv")) {

            writer.write("Title,Description,Due Date,Completed\n");

            for (Task task : tasks) {

                writer.write(task.title + "," + task.description + "," + task.dueDate + "," + (task.isCompleted ? "Yes" : "No") + "\n");

            }

            System.out.println("Tasks saved to tasks.csv");

        } catch (IOException e) {

            System.out.println("Error saving tasks: " + e.getMessage());

        }

    }

    private static void saveTasks() {

        try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(FILE\_NAME))) {

            out.writeObject(tasks);

        } catch (IOException e) {

            System.out.println("Error saving tasks: " + e.getMessage());

        }

    }

    @SuppressWarnings("unchecked")

    private static void loadTasks() {

        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(FILE\_NAME))) {

            Object obj = in.readObject();

            if (obj instanceof List<?>) {

                tasks.addAll((List<Task>) obj);

            }

        } catch (IOException | ClassNotFoundException e) {

            // Ignore if file doesn't exist yet

        }

    }

}

**GITHUB:** [**GITHUB-SARVAN-2187**](https://github.com/sarvan-2187/OOPS-LAB/tree/main/Project%20-%20Task%20Manager)

**FUTURE IMPROVEMENTS:**

* **Task Prioritization → Allow users to set priority levels (High, Medium, Low) and sort tasks accordingly.**
* **Task Categories/Tags → Add labels like “Work,” “Personal,” or custom tags.**
* **Subtasks & Dependencies → Allow users to break down large tasks into smaller subtasks and define dependencies.**
* **Recurring Tasks → Option to set daily, weekly, or monthly recurring tasks.**
* **Time Tracking → Track time spent on each task and generate reports.**
* **API Integration → Expose REST APIs for mobile apps or third-party integrations.**
* **JWT Authentication** → Secure user authentication using JWT.

**TERMINAL OUTPUT:**

